実環境における不確実性や遅延を考慮した学習に関する研究

修士学位論文

実環境における不確実性や遅延を考慮した学習に関する研究

東北大学 大学院情報科学研究科

システム情報科学専攻 篠原研究室

博士課程前期二年の課程

斎藤　淳哉

2012年2月14日

目次

第1章

序論

1.1 背景

現代社会においてロボットは欠かすことのできない存在となっている．自動車や電化製品，精密機器などを組み立てる産業用ロボットや災害現場で活躍するロボット，遙か彼方の惑星を探査するためのロボットなど枚挙に暇がない．

このようなロボットの制御規則をプログラムする方法は大きく分けて2つある．1つはすべての制御規則を開発者がプログラムする方法である．この方法は大変な労力を要する．また，開発者の経験や思い込みによって設計され，品質が低くなる恐れがある．さらに，災害現場などモデル化が難しい環境下で働くロボットを開発する場合，適切な制御規則を設計することは難しい．もう1つの方法は学習によってロボットに自律的に制御規則を獲得させる方法である．ロボットに学習をさせることで，開発者の負担を減らし，また開発者の主観をおさえて適切な制御を行えるようになることが期待される．

このようなロボットの学習に応用されている機械学習の分野に強化学習がある．学習させたい対象をエージェントと呼ぶ．強化学習において，エージェントを取り巻く環境は確率的に変化し，また，直接的には，環境の完全な情報を得られない．ただし，エージェントは次の情報を得ることができる．まず，エージェントは環境の現在の状態を観測することができる．また，エージェントが行動すると状態は確率的に次の状態へと変化し，さらに，状態と行動に応じた報酬が発生する．エージェントはこの状態と報酬を観測できる．エージェントは，このような不確実性をもつ環境とやりとり

する中で，報酬を頼りに試行錯誤し，将来にわたって得られる報酬の総量が大きくなるような行動方策を，自律的に獲得することを目指す．強化学習をロボットの制御規則の獲得に応用する場合，一般的にセンサ情報の組合わせで状態を表現し，意図した制御規則を獲得してくれるような報酬を設定することで環境をモデル化する．

強化学習はこれまでに二足歩行ロボットの制御 [10] やヘリコプターの制御 [1]，ロボットアームの制御 [8] などに応用されている．すでに応用例は多数あるが，しかし，依然として実環境における強化学習にはさまざまな課題もある．本稿ではその 1 つである “遅延” に注目する．

実環境では観測時や行動時に遅延が生じる．例えば，観測時には次のような遅延が生じる．実環境においてセンサ情報を取得するとき，サンプリング周期を 0 にはできないので，環境の状態の変化と認識に遅延が生じる．また，コストの節約や技術的な制限から制御やロボットの学習のための計算機をネットワークを通じて外部に設置することが考えられるが，通信にも時間がかかるので，遅延が生じる．さらに，性能は高いが計算に時間のかかる制御プログラムや強化学習のプログラムを利用したいときも遅延が生じる．行動時には次のような遅延が生じる．ロボットの関節などを動かしたいとき，モータに電流を流してから実際に動き出すまでに遅延が生じる．また観測時と同様にネットワーク越しに制御や学習をするときや，計算に時間のかかるプログラムを利用するときにも遅延が生じる．

このような遅延があることを考慮せずに強化学習を行うことは効率が悪いことが知られている [18, 12]．そこで，本稿では遅延を考慮した強化学習に関する基礎的な問題の定式化と学習アルゴリズムの提案を行う．

1.2 本稿の構成

第 2 章では，一般的な強化学習と既存研究について説明する．第 3 章では，まず，遅延を考慮した強化学習の既存研究について紹介し，次に，これに対する新たな学習アルゴリズムの提案を行う．第 4 章では，新たな問題設定として，遅延の大きさが未知の場合について定式化を行い，学習アルゴリズムの提案を行う．第 5 章では，実験を行い提案手法の有効性を検証する．第 6 章では，まとめと今後の課題を述べる．

第2章

強化学習

この章では，強化学習の一般的な問題設定と学習アルゴリズムについて述べる．はじめに強化学習の問題設定を，次に，環境の一般的な設定である，マルコフ決定過程を説明する．そして，学習アルゴリズムの種類である，モデルベース学習とモデルフリー学習について紹介する．

2.1 強化学習の問題設定

強化学習の目的は，試行錯誤によってエージェントが大きな報酬を得られる行動方策を自律的に獲得することである．エージェントが環境に対して行動を取ると環境の状態が変化する．エージェントは変化した状態と報酬を観測する．これを繰り返すことでエージェントは行動方策を学習していく．なお，本稿では時間と状態と行動は離散で，さらに状態と行動は有限であると仮定する．この仮定の下，強化学習の問題の流れをより詳細に説明する．$\mathbb{N}$ を 0 を含む自然数の集合とする．$\mathbb{R}$ を実数の集合とする．S を状態の集合とする．A を行動の集合とする．ステップ $n \in \mathbb{N}$ における環境とエージェントのやりとりの様子を図 2.1 に示す．試行開始時の状態 $s_0 \in S$ を初期状態と呼ぶ．ステップ n の状態を $s_n \in S$ と表す．ステップ n で，エージェントはそれまでの環境とのやりとりの中で得られた行動方策 $\pi : S \to A$ を用いて，環境に対して行動 $a_n = \pi(s_n)$ を取る．これにより環境の状態が s_n から s_{n+1} に遷移し，報酬 $r_{n+1} \in \mathbb{R}$ が発生する．そしてエージェントは状態 s_{n+1} と報酬 r_{n+1} を観測する．ステップが進み，状態が終端状態になると，現在の試行が終了し新たにステップ 0 から試行が再開する．

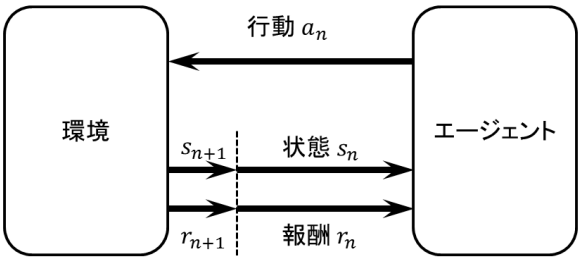

図 2.1: 強化学習における環境とエージェントのやりとり

2.2 マルコフ決定過程

強化学習では，一般的に環境が**マルコフ性**を満たすことを仮定する．環境がマルコフ性を満たすとは，状態が，1 ステップ前の状態と行動のみに依存して確率的に独立に決まることである．マルコフ性を満たす環境のことを**マルコフ決定過程**（*Markov Decision Process : MDP*）と呼ぶ．次のように MDP を定義する．MDP とは $\langle S, A, P, R \rangle$ である．S は状態の集合，A は行動の集合，$P : S \times A \times S \to [0,1]$ は**遷移確率**，$R : S \times A \to \mathbb{R}$ は**報酬関数**である．$P(s'|s,a)$ は，状態 s で行動 a をしたときに状態 s' に遷移する確率を，$R(s,a)$ は，状態 s で行動 a をしたときにエージェントが観測する報酬の期待値を意味する．確率の制限から，P は任意の状態 $s \in S$ と任意の行動 $a \in A$ に対して，$\sum_{s' \in S} P(s'|s,a) = 1$ を満たす．なおエージェントは S と A は与えられるが，P と R を直接観測することはできない．

2.3 モデルベース学習とモデルフリー学習

強化学習において，行動方策を学習する学習アルゴリズムには大きく分けて**モデルベース学習**（*Model-Based Learning*）と**モデルフリー学習**（*Model-Free Learning*）の 2 つがある．それぞれの定義はいくつかあるが [14]，本稿では，モデルベース学習とは，遷移確率と報酬関数を学習して，これらをもとに行動方策を決定する学習アルゴリズム，モデルフリー学習とは，遷移確率と報酬関数を直接学習せず，行動の価値を学習することで行動方策を決定する学習アルゴリズムとする [17]．代表的なモデルベース

学習には，Dyna-Q [16]，E^3 [7]，R-MAX [2] などがある．代表的なモデルフリー学習には，Q-Learning [19]，Q(λ) [19]，Sarsa [11]，Sarsa(λ) [11]，Delayed Q-Learning [14] などがある．モデルベース学習は遷移確率と報酬関数を学習するため，一般的に空間計算量が大きくまたこれらを利用して行動方策を決定するため時間計算量も大きい．これに対しモデルフリー学習は一般的に空間計算量が小さく時間計算量も小さい [13]．近年，モデルベース学習とモデルフリー学習の研究が進み，学習アルゴリズムの収束速度を測る尺度であるサンプル計算量 [5] について両者は行動の集合の大きさと状態の集合の大きさに関して同程度であることがわかってきている [14, 17, 9]．これらをもって直ちにモデルフリー学習がモデルベース学習よりも優れているとはいえないが，モデルフリー学習について研究することは重要である．ここでは代表的なモデルフリー学習の中から Sarsa (λ) を紹介する．アルゴリズム 1 に擬似コードを示す．$Q(s,a)$ は **Q 値**と呼ばれ，状態 s で行動 a をする価値を意味する．ここで価値とは将来にわたって得られると期待される報酬の総量である．Q 値は繰り返し更新されることで徐々に適切な値に収束していく．また，$e(s,a)$ は**適格度トレース**と呼ばれ，学習を効率的に進めるために用いられている．適格度トレースの中でもいくつかの手法が提案されているが，この Sarsa (λ) は，選択されていない行動のトレースを消去するオプションを含んだ入替更新トレースを用いている [15]．Q 値をもとにした代表的な行動方策には，**ε-貪欲法**や**ソフトマックス法**がある．ε-貪欲法は，任意のステップで確率 $1-\varepsilon$ で現在の状態の Q 値が最も大きい行動をとり，確率 ε で一様乱択により行動を決定する．ソフトマックス法は，任意のステップで状態が s のとき，行動 a を次の確率で選択する．

$$\frac{\exp(\beta Q(s,a))}{\sum_{a\in A}\exp(\beta Q(s,a))}$$

β は逆温度と呼ばれるパラメータで，値が大きいほど貪欲な選択を行うようになる．

アルゴリズム 1: Sarsa(λ)

入力: 学習率 α, 割引率 γ, トレース減衰パラメータ λ

初期化:
for 任意の $s, s' \in S$, $a \in A$ に対して **do**
    $Q(s,a) \leftarrow 0$;

for 任意の試行に対して **do**
    for 任意の状態 $s \in S$ と行動 $a \in A$ に対して **do**
        $e(s,a) \leftarrow 0$;
    $s_0 \leftarrow$ 初期状態;
    for 任意のステップ $n \in \mathbb{N}$ に対して **do**
        if $n \geq 2$ **then**
            for 任意の状態 $s \in S$ と行動 $a \in A$ に対して **do**
                if $s = s_{n-2} \wedge a = a_{n-2}$ **then**
                    $e(s,a) \leftarrow 1$;
                else if $s = s_{n-2} \wedge a \neq a_{n-2}$ **then**
                    $e(s,a) \leftarrow 0$;
                else /* $s \neq s_{n-2}$ */
                    $e(s,a) \leftarrow \gamma \cdot \lambda \cdot e(s,a)$;
            $\delta \leftarrow r_{n-1} + \gamma \cdot Q(s_{n-1}, a_{n-1}) - Q(s_{n-2}, a_{n-2})$;
            for 任意の状態 $s \in S$ と行動 $a \in A$ に対して **do**
                $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \delta \cdot e(s,a)$;
            $a_n \leftarrow Q(s_n, a)$ をもとにした行動方策πを用いて s_nで取る行動 $a \in A$ を選ぶ;
        else
            $a_n \leftarrow Q(s_n, a)$ をもとにした行動方策πを用いて s_nで取る行動 $a \in A$ を選ぶ;
        行動 a_n を取り，状態 s_{n+1}，報酬 r_{n+1} を観測する;

第3章

一定の遅延のある環境下での強化学習

この章では，一定の遅延のある場合のマルコフ決定過程に対する強化学習を扱う．はじめに，定遅延マルコフ決定過程を紹介し，次に，これに対する既存手法と提案手法を示す.

3.1 定遅延マルコフ決定過程

状態や報酬の観測時や，行動時に遅延のある MDP に対する強化学習を考える．ただし，遅延は一定で大きさは既知であるとする．状態や報酬の観測時の遅延は**観測遅延**（*Observation Delay*），行動時の遅延は**行動遅延**（*Action Delay*）と呼ばれる．両者はエージェントにとって本質的に同一であることがわかっており [6]，まとめて**制御遅延**（*Control Delay*） [12] と呼ばれる．以降では制御遅延のことを単に**遅延**と呼ぶ. [18] にならい，**定遅延マルコフ決定過程**（*Constant Delayed MDP : CDMDP*）を次のように定義する．CDMDP とは $\langle S, A, P, R, k^* \rangle$ である．S は状態の集合，A は行動の集合，$P : S \times A \times S \to [0,1]$ は遷移確率，$R : S \times A \to \mathbb{R}$ は報酬関数，$k^* \in \mathbb{N}$ は遅延である．ステップ n でエージェントが行動 a_n を取るとき，状態 s_n は確率 $P(s'|s_n, a_{n-k^*})$ にもとづいて状態 s_{n+1} に確率的に独立に遷移する．厳密には次が成り立つと定義する．標本空間を $\Omega = (S \times A)^{\mathbb{N}}$ とし，確率空間を $(\Omega, \Pr)$ とする．$n \in \mathbb{N}$ について，確率変数 $s_n : \Omega \to S$ と $a_n : \Omega \to A$ を

$$\omega = (s^{(0)}, a^{(0)}, s^{(1)}, a^{(1)}, \ldots, s^{(n)}, a^{(n)}, \ldots) \in \Omega$$

に対して，

$$s_n(\omega) = s^{(n)}$$
$$a_n(\omega) = a^{(n)}$$

とする．任意の $n \geq k^*$ と $s^{(0)}, s^{(1)}, \cdots \in S$ と $a^{(0)}, a^{(1)}, \cdots \in A$ について

$$\begin{aligned}
&\Pr\left[s_{n+1} = s^{(n+1)} | s_0 = s^{(0)}, a_0 = a^{(0)}, \ldots, s_{n-k^*} = s^{(n-k^*)}, a_{n-k^*} = a^{(n-k^*)}, \ldots, s_n = s^{(n)}\right] \\
&= \Pr\left[s_{n+1} = s^{(n+1)} | a_{n-k^*} = a^{(n-k^*)}, s_n = s^{(n)}\right] \\
&= P(s^{(n+1)} | s^{(n)}, a^{(n-k^*)})
\end{aligned}$$

を満たすとする．また，報酬 r_{n+1} は期待値 $R(s_n, a_{n-k^*})$ にもとづいて確率的に独立に決まる．ただし $n \leq k^*$ で状態 s_n と報酬 r_n がどのように決定されるかは定義しない．通常の遅延のない MDP では，状態 s_{n+1} と報酬 r_{n+1} は，状態 s_n と行動 a_n のみに依存して確率的に独立に決まり，これを想定して学習アルゴリズムが設計されているので，後述する実験で示すが，CDMDP に対して遅延を考慮していない学習アルゴリズムを使うと，学習の効率は大きく低下する．

3.2 遅延のないマルコフ決定過程への帰着

CDMDP に対する学習アルゴリズムとして，CDMDP を遅延のない通常の MDP へ帰着させ，通常の MDP に対する学習アルゴリズムを用いる手法が提案されている [6]．このアルゴリズムは，行動はそのままで状態の定義を工夫することで CDMDP を MDP へ帰着する．エージェントは，ステップ n で行動 a_n を取り，状態 s_n から状態 s_{n+1} への遷移を観測する．このとき，MDP のステップ n の状態を $(s_n, a_{n-1}, \ldots, a_{n-k^*})$ として，通常の MDP に対する学習アルゴリズムを実行する．この方法で，遅延を考慮しないときよりは学習の効率が上がる場合もあるが，一方で，遅延 k^* に依存して指数的に状態空間が増大する．これによって，学習の効率の低下，計算量の増大を招く．このような課題を解決するための手法がこれまでにいくつか提案されている．

3.3 モデルベース学習を用いた学習アルゴリズム

CDMDP に対する既存手法の中で，モデルベース学習を用いた学習アルゴリズムを紹介する．Walsh らは，遅延 k^* を考慮して状態 s_n と行動 a_{n-k^*} によって状態 s_{n+1} に遷移し報酬 r_{n+1} を獲得したとしてモデルベース学習を実行することで，遷移確率 P と報酬関数 R の推定を行い，さらに行動選択時は，推定した遷移確率 P を用いた最尤推定で k^* ステップ後の状態 s_{n+k^*} を予測し，これをもとに行動を選択するという Model Based Simulation（MBS） [18] を提案した．MBS は効率的に学習が進行する．しかし，モデルベース学習を用いているため計算量が比較的大きい．

3.4 モデルフリー学習を用いた学習アルゴリズム

CDMDP に対する既存手法の中で，モデルフリー学習を用いた学習アルゴリズムを紹介する．Schuitema らは，Sarsa をもとに，遅延 k^* を考慮して状態 s_n と行動 a_{n-k^*} によって状態 s_{n+1} に遷移し報酬 r_{n+1} を獲得したとして Q 値の更新式を修正したアルゴリズム，dSARSA [12] を提案した．また同様にして Sarsa(λ)，Q-Learning，Q(λ) をそれぞれ応用した，dSARSA(λ)，dQ，dQ(λ) を提案した．モデルフリー学習を用いているため計算量が比較的小さい．Schuitema らによると，実験的に dSARSA(λ) が最も優れていたので，以降では，dSARSA(λ) のみに焦点を当てる．

3.5 モデルフリー学習と遷移状態の予測を用いた学習アルゴリズム

Schuitema らの提案した dSARSA(λ) などのアルゴリズムは，Walsh らの提案した MBS と異なり，行動選択時に状態予測を行っていなかった．そのため学習の効率が悪くなっていた．そこで dSARSA(λ) に，MBS と同様の状態予測とこれを用いた行動選択を追加した，$dSARSA(\lambda)_k$ を提案する．擬似コードをアルゴリズム 2 に示す．

Q 値の更新は dSARSA(λ) と同様に行う．$dSARSA(\lambda)_k$ は $k = k^*$ として実行するが，このアルゴリズムは，$k = 0$ とするとアルゴリズム 1 に示した，遅延を考慮していない通常の Sarsa(λ) と一致する．さらに 24 行目の $Q(\hat{s}_{n+k}, a)$ を $Q(s_n, a)$ に変更すると，

dSARSA(λ) とほぼ等しい[1].

このアルゴリズムの最も重要な部分は，22-24 行目である．$\hat{s}_{n+k}$ は，s_n の k ステップ後の予測状態で，行動選択時に $Q(\hat{s}_{n+k}, a)$ をもとにして行動を選ぶ．状態予測は，1 ステップごとに状態を最尤推定を用いて予測することで状態遷移を模倣して行う．この考えは MBS と同じである．

関数 *Predict* に関してその実装を説明する．アルゴリズム中の $Occ_k(s,a,s')$ は，遅延 k を考慮して，状態 s で行動 a をしたことによって遷移した状態が s' であった回数を意味している．遅延 k を考慮して，状態 s で行動 a をしたことによって状態 s' に遷移する確率の見積もりを，Occ_k を用いて，

$$\hat{P}_k(s'|s,a) = \frac{Occ_k(s,a,s')}{\sum_{s' \in S} Occ_k(s,a,s')} \tag{3.1}$$

とする．ここで argmax を定義する．集合 X と X 上の関数 $f : X \to \mathbb{R}$ に対して，

$$M = \{x \in X : \forall y \in X . f(y) \leq f(x)\}$$

とするとき，$\underset{x \in X}{\operatorname{argmax}} f(x)$ を M から 1 つ一様乱択した要素とする．このとき，遅延 k を考慮して状態 s で行動 a をしたことによって遷移すると最尤推定で予測される状態 $\hat{s}$ を

$$\hat{s} = \underset{s' \in S}{\operatorname{argmax}} \hat{P}_k(s'|s,a)$$

とする．関数 *Predict* は，次の関数を $i = 0, \ldots, k-1$ に対して再帰的に適用することで，s_n の k ステップ後の状態を予測する．

$$\hat{s}_{n+(i+1)} = \underset{s' \in S}{\operatorname{argmax}} \hat{P}_k(s' \mid \hat{s}_{n+i}, a_{n-(k-i)})$$

[1] 14–18 行目の 適格度トレース $e_k(s,a)$ の更新式が少し異なるが本質的な問題ではない.

アルゴリズム 2: dSARSA$(\lambda)_k$

入力: 学習率 α, 割引率 γ, トレース減衰パラメータ λ, 遅延 k

初期化:
for 任意の $s, s' \in S,\ a \in A$ に対して **do**
    $Q(s,a) \leftarrow 0$;
    $Occ_k(s,a,s') \leftarrow 0$;

for 任意の試行に対して **do**
    for 任意の状態 $s \in S$ と行動 $a \in A$ に対して **do**
        $e(s,a) \leftarrow 0$;
    $s_0 \leftarrow$ 初期状態;
    for 任意のステップ $n \in \mathbb{N}$ に対して **do**
        if $n \geq k+2$ **then**
            for 任意の状態 $s \in S$ と行動 $a \in A$ に対して **do**
                if $s = s_{n-2} \wedge a = a_{n-k-2}$ **then**
                    $e(s,a) \leftarrow 1$;
                else if $s = s_{n-2} \wedge a \neq a_{n-k-2}$ **then**
                    $e(s,a) \leftarrow 0$;
                else /* $s \neq s_{n-2}$ */
                    $e(s,a) \leftarrow \gamma \cdot \lambda \cdot e(s,a)$;
            $\delta \leftarrow r_{n-1} + \gamma \cdot Q(s_{n-1}, a_{n-k-1}) - Q(s_{n-2}, a_{n-k-2})$;
            for 任意の状態 $s \in S$ と行動 $a \in A$ に対して **do**
                $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \delta \cdot e(s,a)$;
            // 最尤推定による状態の予測のための訓練
            $Occ_k(s_{n-1}, a_{n-k-1}, s_n) \leftarrow Occ_k(s_{n-1}, a_{n-k-1}, s_n) + 1$;
            // 最尤推定による状態の予測
            $\hat{s}_{n+k} \leftarrow Predict(s_n, \{a_{n-k}, \ldots, a_{n-1}\})$;
            $a_n \leftarrow Q(\hat{s}_{n+k}, a)$ をもとにした行動方策πを用いて$\hat{s}_{n+k}$で取る行動 $a \in A$ を選ぶ;
        else
            $a_n \leftarrow Q(s_n, a)$ をもとにした行動方策πを用いて s_nで取る行動 $a \in A$ を選ぶ;
        行動 a_n を取り，状態 s_{n+1}，報酬 r_{n+1} を観測する;

第4章

大きさ未知で一定の遅延のある環境下での強化学習

この章では，前章の定遅延マルコフ決定過程の遅延の大きさが既知であるという仮定を緩めた，未知定遅延マルコフ決定過程を扱う．後述する実験でも示すが，遅延の大きさを間違って定遅延マルコフ決定過程に対する学習アルゴリズムを実行すると，学習の効率は大きく低下する．しかし，実際は遅延が未知でしかも測定が難しい場合がある．ロボットの制御を外部の計算機を用いた強化学習で行うことを考える．ロボットと計算機は通信によって結ばれるため遅延が生じる．ロボットが制作された後，遅延による問題が表面化したとする．あらかじめ遅延を測定できるような機構が備えられていない場合，遅延を測定するためにはロボットを改良する必要がある．もし，強化学習の問題設定そのままに行動と状態のやりとりのみで真の遅延の大きさがわかれば，手間や費用を節約できる．そして，真の遅延の大きさを推定できれば定遅延マルコフ決定過程に対するアルゴリズムを用いて強化学習を行うことができる．さらに，そのような方法で遅延を推定できたとしても厳密には問題が残る．遅延の推定を行うときと強化学習を行うときでは，計算機が実行する処理が異なるため遅延の大きさが変わる可能性がある．また，その他実装上の都合で遅延の大きさが変わる可能性もある．遅延の推定と強化学習を同時に行うことができればそのような問題は生じない．

はじめに，未知定遅延マルコフ決定過程の定義を行い，次に遅延を推定する素朴な手法を示す．さらに，未知定遅延マルコフ決定過程の持つ理論的性質について示し，これを用いた遅延を推定する手法を示す．最後に，遅延の大きさの推定と強化学習を

同時に行う手法を示す.

4.1 未知定遅延マルコフ決定過程

現実の問題で完全に遅延が未知な場合はほとんどない．多くの場合，真の遅延の上界は簡単に得られる．そこで，真の遅延の上界は既知であるとして問題を定義する．**未知定遅延マルコフ決定過程**（Unknown Constant Delayed MDP : UCDMDP）を次のように定義する．UCDMDP とは $\langle S, A, P, R, k^*, k_{\max} \rangle$ である．S は状態の集合，A は行動の集合，$P : S \times A \times S \to [0,1]$ は遷移確率，$R : S \times A \to \mathbb{R}$ は報酬関数，k^* は真の遅延，$k_{\max} \in \mathbb{N}$ は $0 \leq k^* \leq k_{\max}$ を満たす，真の遅延の上界である．エージェントには，P，R，k^* は直接的には与えられず，S，A，$k_{\max}$ が与えられる．ここで $K = \{0, 1, \ldots, k_{\max}\}$，$K^- = K - \{k^*\}$ と記号を定義しておく．

なお，遅延に上界を設けても，任意の UCDMDP に対して真の遅延を推定することは難しい．例えば，状態遷移と報酬が行動にまったく依存せず決まる場合を考えると，エージェントは環境から一方的に情報を受け取っているだけなので，そこに遅延が生じているのか知るすべがない．遅延を知覚するためには，状態遷移または報酬が行動に依存して決定する必要がある．すなわち，遷移確率 P や報酬関数 R に何らかの制約が必要である．

4.2 遅延の大きさを求める素朴な手法

ひとつの状態に確実に留まり続けられる行動と確実に別の状態に遷移することのできる行動をもつ UCDMDP は，素朴な手法で遅延の大きさを求めることができる．この素朴な手法を説明する．はじめに，ひとつの状態に留まり続けられる行動を $k_{\max}$ ステップ繰り返す．次に，確実に別の状態に遷移することのできる行動を 1 回取る．行動を行ってから，状態が遷移したことを観測するまでにかかるステップが遅延の大きさである．しかし，この素朴な手法は条件を満たす行動が存在しない不確実な環境では使うことができない.

4.3 未知定遅延マルコフ決定過程の理論的性質

エージェントが状態に依らず任意のステップで行動の集合 A から独立に確率分布 $P(a)$ に従って行動を乱択する行動方策を取るとき，任意の UCDMDP について成り立つ真の遅延を推定するための手がかりとなる定理を示す．このような行動方策の下，ステップ n で状態 s のときステップ $n+1$ で状態 s' に遷移する確率は，

$$\begin{aligned}
\Pr\left[s_{n+1}=s'|s_n=s\right] &= \frac{\Pr\left[s_{n+1}=s', s_n=s\right]}{\Pr\left[s_n=s\right]} \\
&= \sum_{a\in A} \frac{\Pr\left[s_{n+1}=s', s_n=s, a_{n-k^*}=a\right]}{\Pr\left[s_n=s\right]} \\
&= \sum_{a\in A} \frac{\Pr\left[s_{n+1}=s', s_n=s, a_{n-k^*}=a\right]\Pr\left[a_{n-k^*}=a\right]}{\Pr\left[s_n=s\right]\Pr\left[a_{n-k^*}=a\right]} \\
&= \sum_{a\in A} \frac{\Pr\left[s_{n+1}=s', s_n=s, a_{n-k^*}=a\right]\Pr\left[a_{n-k^*}=a\right]}{\Pr\left[s_n=s, a_{n-k^*}=a\right]} \\
&= \sum_{a\in A} \Pr\left[s_{n+1}=s'|s_n=s, a_{n-k^*}=a\right]\Pr\left[a_{n-k^*}=a\right] \\
&= \sum_{a\in A} P(s'|s,a)P(a)
\end{aligned}$$

となるので，

$$P(s'|s) = \sum_{a\in A} P(s'|s,a)P(a)$$

と表す．次が成り立つ．

定理 1. 任意の *UCDMDP* $\langle S, A, P, R, k^*, k_{\max}\rangle$ に対して，エージェントが任意のステップで行動の集合 A から独立に確率分布 $P(a)$ に従って行動を乱択する行動方策を取るとき，任意の $s, s' \in S$ と $a \in A$ と $n \geq k^*$ と $k \in K^-$ に対して，

$$\Pr\left[s_{n+1}=s'|s_n=s, a_{n-k}=a\right] = P(s'|s)$$

が成り立つ．

証明. 定義より，任意の $n \geq k^*$ と $s^{(0)}, s^{(1)}, \cdots \in S$ と $a^{(0)}, a^{(1)}, \cdots \in A$ について

$$\begin{aligned}
&\Pr\left[s_{n+1} = s^{(n+1)} | s_0 = s^{(0)}, a_0 = a^{(0)}, \ldots, a_{n-k^*} = a^{(n-k^*)}, \ldots, s_n = s^{(n)}\right] \\
&= \Pr\left[s_{n+1} = s^{(n+1)} | a_{n-k^*} = a^{(n-k^*)}, s_n = s^{(n)}\right] \\
&= P(s^{(n+1)} | s^{(n)}, a^{(n-k^*)})
\end{aligned}$$

さらに任意の $n \geq 0$ と $s^{(0)}, s^{(1)}, \cdots \in S$ と $a^{(0)}, a^{(1)}, \cdots \in A$ について

$$\begin{aligned}
&\Pr\Big[a_n = a^{(n)} | s_0 = s^{(0)}, a_0 = a^{(0)}, \ldots, s_n = s^{(n)}, s_{n+1} = s^{(n+1)}, a_{n+1} = a^{(n+1)}, \ldots, \\
&\qquad\qquad s_{n+k^*} = s^{(n+k^*)}, a_{n+k^*} = a^{(n+k^*)}, a_{n+k^*+1} = a^{(n+k^*+1)}, \ldots\Big] \\
&= \Pr\left[a_n = a^{(n)}\right] \\
&= P(a^{(n)})
\end{aligned}$$

が成り立つ．よって，任意の $s, s' \in S$，$a \in A$，$n \geq k^*$，$k \in K^-$ に対して，

$$\begin{aligned}
&\Pr\left[s_{n+1} = s' | a_{n-k} = a, s_n = s\right] \\
&= \frac{\Pr\left[s_{n+1} = s', a_{n-k} = a, s_n = s\right]}{\Pr\left[a_{n-k} = a, s_n = s\right]} \\
&= \frac{\sum_{a' \in A} \Pr\left[s_{n+1} = s', a_{n-k} = a, s_n = s, a_{n-k^*} = a'\right]}{\Pr\left[a_{n-k} = a, s_n = s\right]} \\
&= \frac{\sum_{a' \in A} \Pr\left[s_{n+1} = s' | a_{n-k} = a, s_n = s, a_{n-k^*} = a'\right] \Pr\left[a_{n-k} = a, s_n = s, a_{n-k^*} = a'\right]}{\Pr\left[a_{n-k} = a, s_n = s\right]} \\
&= \frac{\sum_{a' \in A} \left(\begin{array}{l} \Pr\left[s_{n+1} = s' | a_{n-k} = a, s_n = s, a_{n-k^*} = a'\right] \\ \times \Pr\left[a_{n-k^*} = a' | a_{n-k} = a, s_n = s\right] \Pr\left[a_{n-k} = a, s_n = s\right] \end{array} \right)}{\Pr\left[a_{n-k} = a, s_n = s\right]} \\
&= \frac{\sum_{a \in A} \Pr\left[s_{n+1} = s' | s_n = s, a_{n-k^*} = a'\right] \Pr\left[a_{n-k^*} = a'\right] \Pr\left[a_{n-k} = a, s_n = s\right]}{\Pr\left[a_{n-k} = a, s_n = s\right]} \\
&= \sum_{a' \in A} \Pr\left[s_{n+1} = s' | s_n = s, a_{n-k^*} = a'\right] \Pr\left[a_{n-k^*} = a'\right] \\
&= \sum_{a' \in A} P(s' | s, a') P(a') \\
&= P(s' | s)
\end{aligned}$$

が成り立つ． □

アルゴリズム 3: CPAS

入力: 遅延の上界 $k_{\max}$，状態 $s^{\#}$，訓練回数 θ，テスト回数 σ

// 状態の集合を $S=\{s^1,s^2,\ldots,s^u\}$，行動の集合を $A=\{a^1,a^2,\ldots,a^v\}$ とする

// エージェントは任意のステップで行動の集合 A から一様乱択する行動方策を取る

// ステップ n の状態を s_n，そのとき取る行動を a_n とする

$n \leftarrow k_{\max}$;
for 任意の $k \in K$ に対して **do**
  for 任意の $j \in \{1,2,\ldots,v\}$ に対して **do**
    $c \leftarrow 1$;
    repeat
      if $s_n = s^{\#} \wedge a_{n-k} = a^j$ **then**
        for 任意の $i \in \{1,2,\ldots,u\}$ に対して **do**
          if $s_{n+1} = s^i$ **then**
            $X_c^{i,j,k} \leftarrow 1$;
          else
            $X_c^{i,j,k} \leftarrow 0$;
          $c \leftarrow c+1$;
      $n \leftarrow n+1$;
    until $c > \theta$;
    for 任意の $i \in \{1,2,\ldots,v\}$ に対して **do**
      // 状態 $s^{\#}$ で行動 a^j をしたことによって，状態 s^i に遷移する確率の見積もり
      $\hat{P}_k(s^i|s^{\#},a^j) \leftarrow (X_1^{i,j,k}+X_2^{i,j,k}+\cdots+X_\theta^{i,j,k})/\theta$;
for 任意の $k \in K$ に対して **do**
  for 任意の $j \in \{1,2,\ldots,v\}$ に対して **do**
    $d \leftarrow 1$;
    repeat
      if $s_n = s^{\#} \wedge a_{n-k} = a^j$ **then**
        if $s_{n+1} = \underset{i\in\{1,2,\ldots,u\}}{\operatorname{argmax}} \hat{P}_k(s^i|s^{\#},a^j)$ **then**
          $Y_d^{j,k} \leftarrow 1$;
        else
          $Y_d^{j,k} \leftarrow 0$;
        $d \leftarrow d+1$;
      $n \leftarrow n+1$;
    until $d > \sigma$;
    $T^{j,k} \leftarrow (Y_1^{j,k}+Y_2^{j,k}+\cdots+Y_\sigma^{j,k})/\sigma$; // 遅延 k を想定したときの行動 a^j に対する正解率
  $T^k \leftarrow (T^{1,k}+T^{2,k}+\cdots+T^{v,k})/v$; // 遅延 k を想定したときの正解率
return $\underset{k\in K}{\operatorname{argmax}}\, T^k$;

定理 1 を利用すると，素朴な手法では遅延を推定することができない不確実な環境でも真の遅延を推定することができる．定理 1 を利用して真の遅延を推定するアルゴリズム，*CPAS*（*Comparison of Prediction Accuracy of States*）を提案する．CPAS の疑似コードをアルゴリズム 3 に示す．このアルゴリズムは，まず，$0,1,\ldots,k_{\max}$ の遅延をそれぞれ想定して，ステップ n の状態 s_n がある状態 $s^{\#} \in S$ であったときに，この状態で行動 a_{n-k} をしたことによって状態 s_{n+1} に遷移する確率を計算し，状態 s_{n+1} を最尤推定する分類器をつくる．そして，想定している遅延ごとに次を行う．状態 s_n が状態 $s^{\#}$ でかつ行動 a_{n-k} が行動 $a \in A$ だったときに次の状態 s_{n+1} を予測することでテストを行い，遅延 k を想定したときの行動 a に対する正解率を計算する．同様にすべての行動ごとに正解率を計算し，これらの平均を取って，遅延 k を想定したときの正解率を計算する．以上を想定している遅延ごとに行い，遷移確率 P がある条件を満たすとき，真の遅延を想定したときの正解率は間違った遅延を想定したときの正解率よりも大きくなることを利用して真の遅延を推定する．log を自然対数とする．CPAS について次が成り立つ．

定理 2. *UCDMDP* $\langle S,A,P,R,k^*,k_{\max}\rangle$ は状態 $s^{\#} \in S$ と定数 $0 < \tau \leq 1$ と定数 $0 < \varepsilon \leq 1$ について，任意の $a \in A$ に対して，

$$\left|\{s' \in S : \max_{s' \in S} P(s'|s^{\#},a) = P(s'|s^{\#},a)\}\right| = 1 \tag{4.1}$$

と，任意の $s \in S$ と任意の $a \in A$ に対して，

$$\max_{s' \in S} P(s'|s^{\#},a) \neq P(s|s^{\#},a) \Rightarrow \max_{s' \in S} P(s'|s^{\#},a) - P(s|s^{\#},a) \geq \tau \tag{4.2}$$

と，

$$\frac{1}{|A|}\sum_{a \in A} \max_{s' \in S} P(s'|s^{\#},a) - \frac{1}{|A|} \max_{s' \in S} \sum_{a \in A} P(s'|s^{\#},a) \geq \varepsilon \tag{4.3}$$

を満たすとする．

任意の $0 < \delta \leq 1$ について，

$$\rho = \log \frac{2\max(|S||A|,(k_{\max}+1)|A|)}{\delta}$$

とし，訓練回数を $\theta = \left\lceil \frac{2}{\tau^2}\rho \right\rceil$，テスト回数を $\sigma = \left\lceil \frac{2}{\varepsilon^2}\rho \right\rceil$ として，この *UCDMDP* の真の遅延を *CPAS* で推定するとき，*CPAS* は，停止するならば，確率 $1-\delta$ 以上で真の遅延を返す.

証明のための定理を用意する.

定理 3 (Hoeffding の不等式 [4]). $[0,1]$ の範囲の値をとる，独立な確率変数 $X^1, X^2, \ldots, X^n$ の和を $X = \sum_{i=1}^{n} X^i$ とすると，任意の $\varepsilon > 0$ に対して，

$$\Pr\left[\left|\frac{X}{n} - \mathrm{E}\left[\frac{X}{n}\right]\right| \geq \varepsilon\right] \leq 2\exp(-2\varepsilon^2 n)$$

が成り立つ.

証明. 定理 2 を証明する．擬似コード内の記号を用いる． $I = \{1,2,\ldots,u\}$，$J = \{1,2,\ldots,v\}$ とする．任意の $c \in \{1,2,\ldots,\theta\}$ と $i \in I$ と $j \in J$ と $k \in K$ について，CDMDP の定義と定理 1 より，

$$\mathrm{E}\left[X_c^{i,j,k}\right] = \begin{cases} P(s^i|s^\#,a^j) & \text{if } k = k^* \\ P(s^i|s^\#) & \text{otherwise} \end{cases}$$

が成り立つ．任意の $i \in I$ と $j \in J$ について，

$$\left|\hat{P}_{k^*}(s^i|s^\#,a^j) - P(s^i|s^\#,a^j)\right| < \frac{1}{2}\tau \tag{4.4}$$

であれば，式 (4.2) より常に，任意の $j \in J$ について，

$$\operatorname*{argmax}_{i\in I} \hat{P}_{k^*}(s^i|s^\#,a^j) = \operatorname*{argmax}_{i\in I} P(s^i|s^\#,a^j) \tag{4.5}$$

が成り立つ．Hoeffding の不等式より，ひとつの $i \in I$ とひとつの $j \in J$ について，式 (4.4) が成り立つ確率は，少なくとも $1-2\exp(-\frac{1}{2}\tau^2\theta)$ である．よって，任意の $i \in I$ と任意の $j \in J$ に対して式 (4.4) が成り立つ確率は，少なくとも $1-2|S||A|\exp(-\frac{1}{2}\tau^2\theta)$ である．したがって，式 (4.5) が任意の $j \in J$ に対して成り立つ確率は，少なくとも

$$1-2|S||A|\exp(-\frac{1}{2}\tau^2\theta) \tag{4.6}$$

である.

式 (4.5) が任意の $j \in J$ に対して成り立つとき，$k = k^*$ で 22 行目の等号が成立する確率は，CDMDP の定義より，任意の $n \geq k_{\max}$ と $j \in J$ について，

$$\begin{aligned}\Pr\left[s_{n+1} = \operatorname*{argmax}_{i \in I} P(s^i|s^\#, a^j) \,\middle|\, s_n = s^\#, a_{n-k} = a^j\right] &= P\left(\operatorname*{argmax}_{i \in I} P(s^i|s^\#, a^j) \,\middle|\, s^\#, a^j\right) \\ &= \max_{i \in I} P\left(s^i \,\middle|\, s^\#, a^j\right)\end{aligned}$$

である．したがって，式 (4.5) が任意の $j \in J$ に対して成り立つとき，任意の $d \in \{1, 2, \ldots, \sigma\}$ と $j \in J$ について，

$$\mathrm{E}\left[Y_d^{j,k^*}\right] = \max_{i \in I} P(s^i|s^\#, a^j) \tag{4.7}$$

が成り立つ．

$\mathcal{F}$ を $A \to S$ のすべての集合とし，22 行目の $\operatorname*{argmax}_{i \in I} \hat{P}_k(s^i|s^\#, \cdot)$ を $f \in \mathcal{F}$ で表す．任意の $n \geq k_{\max}$ と $k \in K^-$ と $j \in J$ について，定理 1 より，

$$\begin{aligned}\max_{f \in \mathcal{F}} \Pr\left[s_{n+1} = f(a^j) \,\middle|\, s_n = s^\#, a_{n-k} = a^j\right] &= \max_{f \in \mathcal{F}} P\left(f(a^j) \,\middle|\, s^\#\right) \\ &= \max_{i \in I} P\left(s^i \,\middle|\, s^\#\right)\end{aligned}$$

が成り立つ．よって，任意の $d \in \{1, 2, \ldots, \sigma\}$ と $j \in J$ と $k \in K^-$ について，

$$\mathrm{E}\left[Y_d^{j,k}\right] \leq \max_{i \in I} P(s^i|s^\#) \tag{4.8}$$

が成り立つ．

したがって，式 (4.5) が任意の $j \in J$ に対して成り立つとき，式 (4.7) より，任意の $j \in J$ について，

$$\mathrm{E}\left[T^{j,k^*}\right] = \max_{i \in I} P(s^i|s^\#, a^j) \tag{4.9}$$

が成り立ち，式 (4.8) より，任意の $j \in J$ と $k \in K^-$ について，

$$\mathrm{E}\left[T^{j,k}\right] \leq \max_{i \in I} P(s^i|s^\#) \tag{4.10}$$

が成り立つ．

ひとつの $k \in K$ とひとつの $j \in J$ について，

$$\left| T^{j,k} - \mathrm{E}\left[T^{j,k} \right] \right| < \frac{1}{2}\varepsilon \tag{4.11}$$

となる確率は，Hoeffding の不等式より，少なくとも $1 - 2\exp(-\frac{1}{2}\varepsilon^2\sigma)$ である．よって任意の $k \in K$ と $j \in J$ について，式 (4.11) が成り立つ確率は，少なくとも

$$1 - 2(k_{\max} + 1)|A| \exp(-\frac{1}{2}\varepsilon^2\sigma) \tag{4.12}$$

である．

また，式 (4.5) が任意の $j \in J$ に対して成り立つとき，式 (4.9) より，

$$\begin{aligned} \mathrm{E}\left[T^{k^*} \right] &= \frac{1}{|A|} \sum_{j \in J} \mathrm{E}\left[T^{j,k^*} \right] \\ &= \frac{1}{|A|} \sum_{j \in J} \max_{i \in I} P(s^i | s^{\#}, a^j) \end{aligned} \tag{4.13}$$

が成り立ち，式 (4.10) より，$k \in K^-$ について，

$$\begin{aligned} \mathrm{E}\left[T^{k} \right] &= \frac{1}{|A|} \sum_{j \in J} \mathrm{E}\left[T^{j,k} \right] \\ &\leq \frac{1}{|A|} \sum_{j \in J} \max_{i \in I} P(s^i | s^{\#}) \\ &= \max_{i \in I} P(s^i | s^{\#}) \\ &= \frac{1}{|A|} \max_{i \in I} \sum_{j \in J} P(s^i | s^{\#}, a^j) \end{aligned} \tag{4.14}$$

が成り立つ．任意の $k \in K$ と $j \in J$ について式 (4.11) が成り立つとき，任意の $k \in K$ について，常に，

$$\left| T^{k} - \mathrm{E}\left[T^{k} \right] \right| < \frac{1}{2}\varepsilon$$

が成り立つ．これと，さらに式 (4.5) が任意の $j \in J$ に対して成り立つとき，式 (4.3) と式 (4.13) と式 (4.14) より，任意の $k \in K^-$ について

$$T^{k^*} > T^{k}$$

が成り立つ．このときアルゴリズムは真の遅延を常に返す.

以上の議論から，アルゴリズムが真の遅延を返さない確率は，式 (4.6) と式 (4.12) より，高々

$$2|S||A|\exp(-\frac{1}{2}\tau^2\theta)+2(k_{\max}+1)|A|\exp(-\frac{1}{2}\varepsilon^2\sigma)$$

である．この確率を δ で抑えられれば，アルゴリズムが真の遅延を返す確率は少なくとも $1-\delta$ であるといえる．ρ を

$$\theta \geq \frac{2}{\tau^2}\rho$$
$$\sigma \geq \frac{2}{\varepsilon^2}\rho$$

を満たす値とするとき次のように式を整理する.

$$\begin{aligned}
&2|S||A|\exp(-\frac{1}{2}\tau^2\theta)+2(k_{\max}+1)|A|\exp(-\frac{1}{2}\varepsilon^2\sigma) \leq \delta \\
&2|S||A|\exp(-\rho)+2(k_{\max}+1)|A|\exp(-\rho) \leq \delta \\
&2\max\left(|S||A|,(k_{\max}+1)|A|\right)\exp(-\rho) \leq \delta \\
&\rho \geq \log\frac{2\max\left(|S||A|,(k_{\max}+1)|A|\right)}{\delta}
\end{aligned}$$

したがって，

$$\begin{aligned}
\theta &= \left\lceil \frac{2}{\tau^2}\rho \right\rceil \\
\sigma &= \left\lceil \frac{2}{\varepsilon^2}\rho \right\rceil \\
\rho &= \log\frac{2\max\left(|S||A|,(k_{\max}+1)|A|\right)}{\delta}
\end{aligned}$$

とするとき，CPAS は，停止するならば，確率 $1-\delta$ 以上で真の遅延を返す.

□

定理 2 について議論する．式 (4.2) と式 (4.3) の条件に関して，任意の遷移確率 P について，任意の $s \in S$ と任意の $a \in A$ に対して，

$$\max_{s' \in S} P(s'|s^{\#},a) \neq P(s|s^{\#},a) \Rightarrow \max_{s' \in S} P(s'|s^{\#},a) - P(s|s^{\#},a) \geq \tau$$

を満たすような $0 < \tau \leq 1$ が必ず存在し，また，

$$\frac{1}{|A|}\sum_{a\in A}\max_{s'\in S}P(s'|s^{\#},a) - \frac{1}{|A|}\max_{s'\in S}\sum_{a\in A}P(s'|s^{\#},a) \geq 0$$

は常に成り立つことから，これらの条件は緩い制約といえる．なお式 (4.3) の条件は，状態遷移が行動に依存する必要があることを意味している．また式 (4.1) の条件は適切な証明でおそらく緩められると思われる．しかし停止性の条件は比較的厳しい．停止するためにはまず，任意のステップで行動の集合 A から一様乱択する行動方策を取って，状態 $s^{\#}$ に少なくとも $|A|(k_{\max}+1)(\theta+\sigma)$ 回到達しなければならない．さらに到達したステップの $k \in K$ ステップ前の行動がアルゴリズムの求める行動でなければならない．よって CPAS は理論的な保証が得られるものの，有限ステップで停止する場合でもかなりのステップを要するため実用的ではない．そこで次節では，真の遅延を高速に推定しさらに強化学習を同時に行う，実用的なアルゴリズムを提案する．

4.4 遷移状態の予測の正解率を用いた学習アルゴリズム

前節では，遷移状態の予測の正解率を使って真の遅延を推定するアルゴリズムを示した．真の遅延を正しく推定できれば，推定した遅延をもとに CDMDP に対する強化学習アルゴリズムを実行することができる．しかし，遅延の推定を行うときと強化学習を行うときで遅延の大きさが異なる可能性もあるので同時に行うほうがより適切である．そこで本節では，dSARSA$(\lambda)_k$ を応用し，遷移状態の予測の正解率を使った，*dSARSA*$(\lambda)_X$ を提案する．dSARSA$(\lambda)_X$ は 1 個のマスタと $k_{\max}+1$ 個のスレイブから構成される．スレイブは $0,1,\ldots,k_{\max}$ からそれぞれ遅延を想定し，任意のステップ n で，任意の状態 s_n と行動 a_{n-k} から状態 s_{n+1} に遷移するとして学習し，予測の正解率を求め，さらに Q 値の学習によってこれを利用した適切な行動選択を行う．そしてスレイブはマスタに，選択した行動と，スレイブの選択した行動が適切であると信頼できる度合，すなわち信頼度として正解率を通知する．マスタは任意のステップで信頼度の最も大きいスレイブの選んだ行動を実際に取る．

dSARSA$(\lambda)_k$ は，CPAS のように任意のステップで行動の集合 A から行動を一様乱択する行動方策でないので，真の遅延を想定したスレイブの信頼度が最も大きくなる

アルゴリズム 4: dSARSA(λ)$_X$: マスタ

入力: 学習率 α, 割引率 γ, 適格度トレースパラメータ λ, 遅延の上限 $k_{\max}$

```
1  初期化:
2  for k ∈ K do
3  |   α，γ，λ，k を用いてスレイブを生成;
4
5  for 任意の試行に対して do
6  |   s_0 ← 初期状態;
7  |   for 任意のステップ n ∈ ℕ に対して do
8  |   |   すべてのスレイブから行動と信頼度を受け取る;
9  |   |   信頼度が最大のスレイブの選んだ行動 a_n を取り，
   |   |   状態 s_{n+1} と報酬 r_{n+1} を観測する;
10 |   |   すべてのスレイブに行動 a_n，状態 s_{n+1}，報酬 r_{n+1} を渡す;
```

ことを保証することはできないが，後述する実験では正確かつ高速に真のスレイブの信頼度が最も大きくなった．すなわち，真の遅延を正確かつ高速に推定できた．

このアルゴリズムは，Hedge(β) [3] のようなオンライン割り当て問題に対するアルゴリズムと捉えることができる．また別の見方をすれば，分類器の学習におけるメタパラメータの調整をオンラインで並列に行っているとも考えられる．

マスタの擬似コードをアルゴリズム 4 に示す．スレイブはアルゴリズム 2 で示した dSARSA(λ)$_k$ の擬似コードを以下で一部修正することで説明する．

1. 10 行目の前に挿入する:

$$t \leftarrow 0;$$

$$T \leftarrow 0;$$

$$conf_0 \leftarrow 0;$$

T は予測が正解しているかテストした回数で，t はそのうちで正解した回数である．

2. 22 行目の前に挿入する:

$$\hat{s}_n \leftarrow Predict(s_{n-1}, \{a_{n-k-1}\});$$
$$\textbf{if } \hat{s}_n = s_n \textbf{ then } t \leftarrow t+1;$$
$$T \leftarrow T+1;$$
$$conf_n \leftarrow \frac{t}{T};$$

$conf_n$ は信頼度である.

3. 24 行目と 26 行目をそれぞれ入れ替える:

$\hat{a} \leftarrow Q(\hat{s}_{n+k}, a)$ をもとにした行動方策πを用いて$\hat{s}_{n+k}$で取る行動 $a \in A$ を選ぶ;

$\hat{a} \leftarrow Q(s_n, a)$ をもとにした行動方策πを用いて s_nで取る行動 $a \in A$ を選ぶ;

$\hat{a}$ はスレイブがマスタに提案する行動で，実際にマスタが選ぶ行動 a_n とは限らないのでこのように表現する.

4. 27 行目を入れ替える

行動 $\hat{a}$ と信頼度 $conf_n$ をマスタに渡す;
行動 a_n，状態 s_{n+1}，報酬 r_{n+1} をマスタから受け取る;

なお，このアルゴリズムの考えは Sarsa(λ) 以外のモデルフリー学習やモデルベース学習にも応用することができる．ただし， dSARSA$(\lambda)_X$ は $k_{\max}$ に比例して計算量が増大するので計算量の比較的小さいモデルフリー学習のほうが適切である.

第5章

実験

この章では，W-迷路と崖っぷち問題を環境として dSARSA(λ)$_k$ と dSARSA(λ)$_X$ が効率的に報酬を獲得できるか実験を行う．比較のため，次の 3 つのアルゴリズムについても同様に実験を行う．

(1) Sarsa(λ)（$k = 0$ とした dSARSA(λ)$_k$ ）

(2) dSARSA(λ) [12]（dSARSA(λ)$_k$ の擬似コードの 24 行目の $Q(\hat{s}_{n+k^*}, a)$ を $Q(s_n, a)$ に変更したアルゴリズム）

(3) MBS+R-max [18]（MBS にモデルベース学習である R-max [2] を組み合わせたアルゴリズム．このアルゴリズムは素朴な実装では非常に計算時間がかかるので，崖っぷち問題に対して実験することはできなかった.）

それぞれの環境でノイズがない場合とある場合の実験を行う．

5.1 W-迷路

はじめに，図 5.1 で表される ***W*-迷路**について実験する．W-迷路は，[18] で提案され，[12] で拡張された問題である．一般的な迷路問題と同じく，エージェントは自己位置を状態として観測する．エージェントが初期位置から目標位置に移動するまでを 1 試行とする．初期位置は迷路上から一様に乱択され，ステップ $n \leq k+1$ では初期位置に留まるとする．目標位置は “GOAL” と印されたところである．エージェントは，ステップごとに行動を「上」「下」「右」「左」の中から選択し移動する．ただし，壁

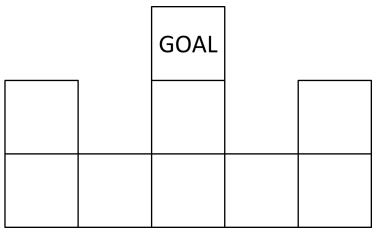

図 5.1: W-迷路 [18]

方向に移動する行動が選ばれた場合は，その位置に留まる．エージェントは任意のステップで報酬 -1 を得る．

5.1.1 ノイズのない場合

ノイズがない場合は，エージェントは選んだ行動の通りに移動することができる．

各学習アルゴリズムのメタパラメータを次のように設定した．Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の学習率を $\alpha = 1.0$，割引率を $\gamma = 0.5$，適格度トレースパラメータを $\lambda = 0.5$ とした．これらは遅延がないときに Sarsa(λ) が効率よく学習できることから選んだパラメータである．Sarsa(λ)， dSARSA$(\lambda)_k$， dSARSA$(\lambda)_X$ の行動方策は探索なしで貪欲に価値の高い行動を選ぶとした．dSARSA(λ) は事前の実験でこの方策では非常に悪い性能を示すことがわかったので，ソフトマックス法を選び，逆温度を $\beta = 0.1$ とした．また， dSARSA(λ) については，[12] で学習率 α を小さくすべきだとしていたので，$\alpha = 0.1$ についても実験した．MBS+R-max のメタパラメータは，実質 R-max のメタパラメータであるが，$T = 10$，$K_1 = 1$ とした．真の遅延を $k^* = 5$ とし，dSARSA$(\lambda)_X$ については遅延の上限を $k_{\max} = 15$ とした．100 試行を 10 回行い，試行ごとの累積の報酬の平均を求めた．

図 5.2，図 5.3 に実験結果を示した．比較のため，理想的な結果として遅延がないときの Sarsa(λ) についても結果を示した．図 5.2 より，Sarsa(λ) や dSARSA(λ) はよい結果が得られなかったのに対して，MBS+R-max と $k = 5$ の dSARSA$(\lambda)_k$ は理想に近い性能を示した．このことは，dSARSA$(\lambda)_k$ で行った状態予測に学習を高速化する効果があったことを意味する．

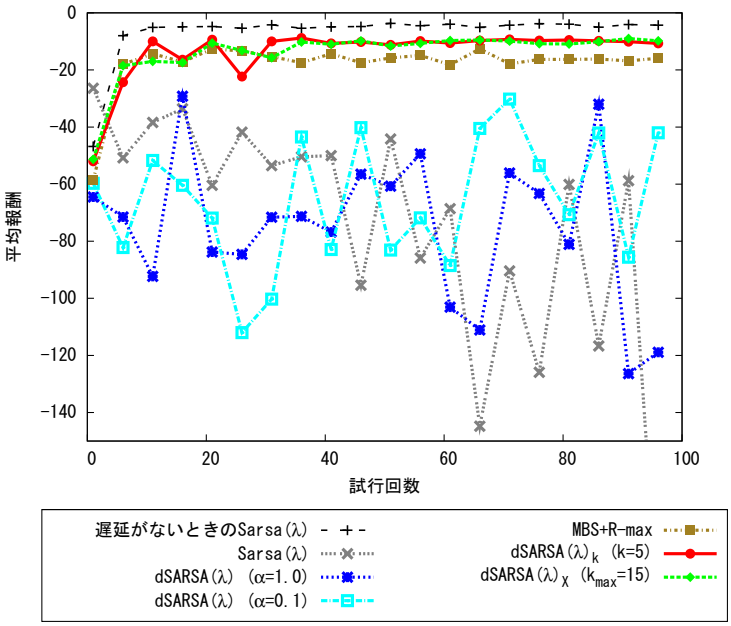

図　5.2: W-迷路を使った実験における，遅延がないときの Sarsa(λ) と，Sarsa(λ), dSARSA(λ)，MBS+R-max，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の比較

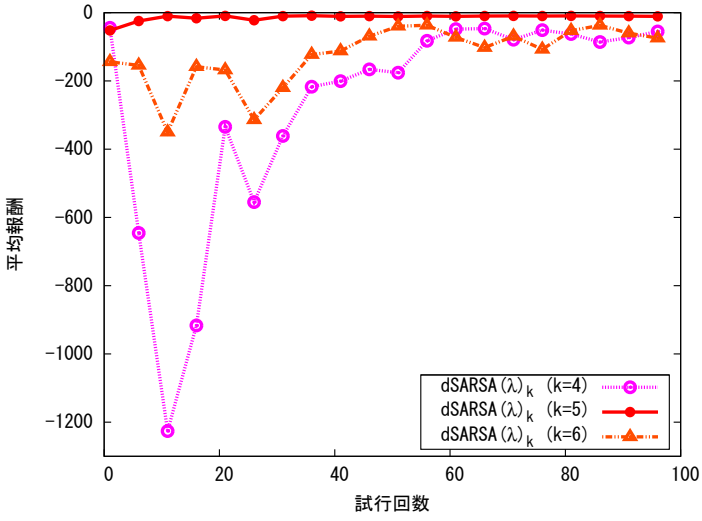

図　5.3: W-迷路を使った実験における，真の遅延 $k=5$ と間違った遅延 $k=4$，$k=6$ をそれぞれ設定したときの dSARSA$(\lambda)_k$ の比較

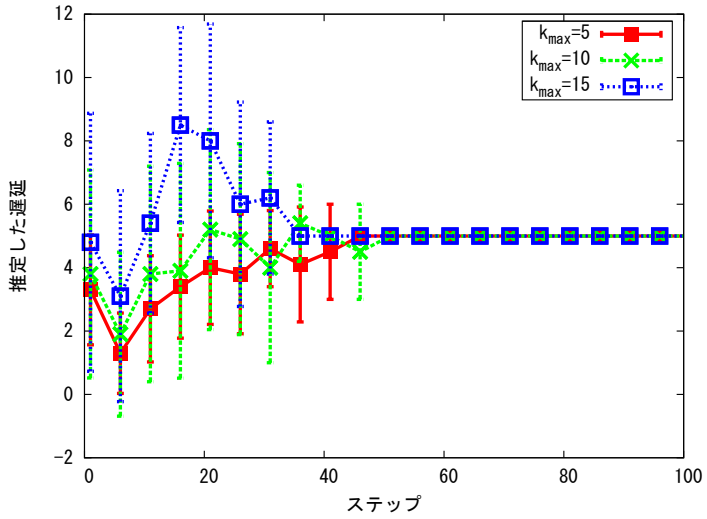

図 5.4: W-迷路を使った実験における，$k_{\max}$ を変化させたときの dSARSA$(\lambda)_X$ が推定した遅延の平均と標準偏差

次に，dSARSA$(\lambda)_k$ で遅延 k を変えたときの性能を検証し，図 5.3 に示した．真の遅延 $k=5$ の場合と比較して，間違った遅延 $k \neq 5$ で学習すると性能は明らかに低下する．ここで，図 5.2 の dSARSA$(\lambda)_X$ の結果をみると，真の遅延を与えられないにもかかわらず，真の遅延を与えられた dSARSA$(\lambda)_k$ と同等の性能を示していることがわかる．

さらに，dSARSA$(\lambda)_X$ に $k_{\max}=5,10,15$ をそれぞれ与えて 10 回実行したとき，信頼度が最も大きくなったスレイブの想定していた遅延，すなわち推定した遅延の平均と標準偏差を図 5.4 に示す．図 5.2，図 5.3 の X 軸は試行回数であったが，図 5.4 ステップである．図 5.2 で $k_{\max}=15$ を与えられた dSARSA$(\lambda)_X$ の最初の試行の平均報酬は -51.3 となっていて，これは，最初の試行のステップ数が平均で 51.3 であることを意味している．これに対して遅延の推定は，どの $k_{\max}$ に対しても約 50 ステップまでに正しく収束していることから，dSARSA$(\lambda)_X$ は正確かつ高速に遅延を推定できていることがわかる．

5.1.2 ノイズのある場合

ノイズがある場合は，エージェントは選んだ行動の通りに移動することができるとは限らない．エージェントは確率 0.7 で選んだ通りに移動できるが，それぞれ確率 0.1 で他の方向へ移動する．なお，このようなノイズのある環境において，遅延の大きさを求める素朴な手法は使うことができない．

各学習アルゴリズムのメタパラメータを次のように設定した．Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の学習率を $\alpha = 0.1$，割引率を $\gamma = 0.5$，適格度トレースパラメータを $\lambda = 0.5$ とした．また，dSARSA(λ) については，$\alpha = 0.01$ についても実験した．Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の行動方策は探索なしで貪欲に価値の高い行動を選ぶとした．MBS+R-max のメタパラメータは，$T = 10$，$K_1 = 10$ とした．真の遅延を $k^* = 5$ とし，dSARSA$(\lambda)_X$ については遅延の上限を $k_{\max} = 15$ とした．100 試行を 10 回行い，試行ごとの累積の報酬の平均を求めた．

図 5.5，図 5.6，図 5.7 に実験結果を示した．それぞれノイズがない場合の実験結果である，図 5.2，図 5.3，図 5.4 に対応している．

ノイズのためにより難しい問題になっているので，遅延がない場合の Sarsa(λ) の累積報酬の平均に達することのできたアルゴリズムはなかったが，その中でも，MBS+R-max，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ は，他に比べて高い性能を示した．また dSARSA$(\lambda)_X$ は，やはりノイズがない場合には及ばないものの，正確かつ高速に遅延を推定した．

5.2 崖っぷち問題

図 5.8 に表した新しい問題設定，**崖っぷち問題**について実験する．エージェントは右端の崖っぷちを目指して駆け上ると高い報酬を得られる．しかし，右に進みすぎると崖から落ちて元いた場所に戻ってしまう．エージェントは，自己位置を状態 $s^1, s^2, \ldots, s^h$ として観測する．初期位置は s^1 で，ステップ $n \leq k+1$ では初期位置に留まるとする．エージェントは各ステップで「左」「停止」「右」から行動を選んで移動する．ただし s^h で右に移動すると左端の初期位置に戻される．エージェントは，状態 s^i にいるときに，報酬 $+i$ を獲得する．本稿では $h = 20$ とした．この問題は，もし遅延がなければ簡単であるが，遅延がある場合は，正確に遅延を予測しないと崖から落ちてしまうた

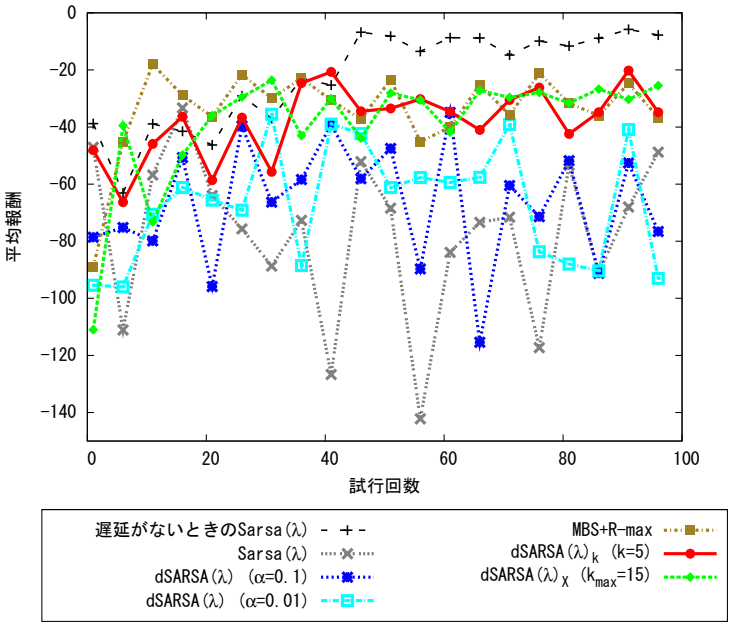

図 5.5: ノイズのある W-迷路を使った実験における，遅延がないときの Sarsa(λ) と，Sarsa(λ)，dSARSA(λ)，MBS+R-max，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の比較

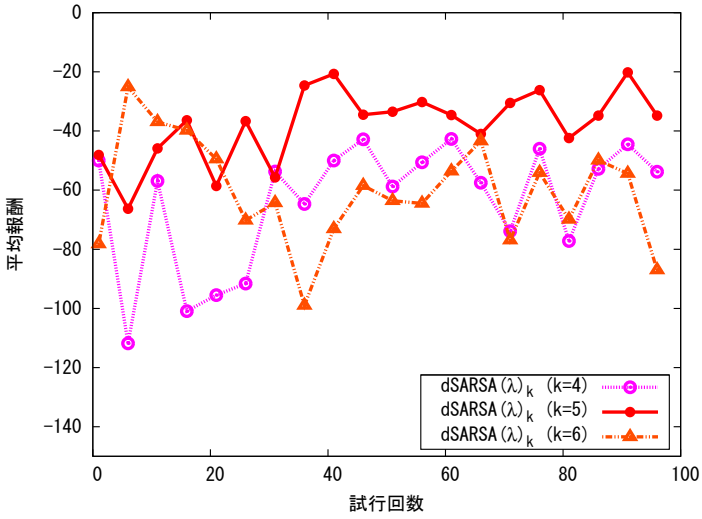

図 5.6: ノイズのある W-迷路を使った実験における，真の遅延 $k=5$ と間違った遅延 $k=4$，$k=6$ をそれぞれ設定したときの dSARSA$(\lambda)_k$ の比較

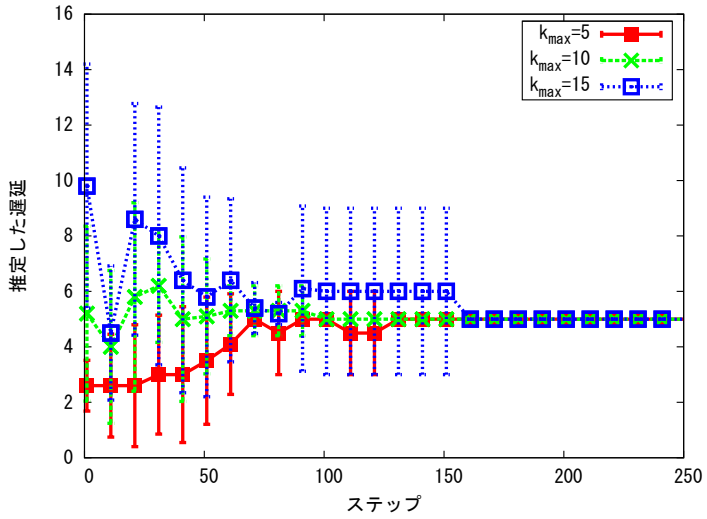

図　5.7:　ノイズのある W-迷路を使った実験における，$k_{\max}$ を変化させたときの dSARSA$(\lambda)_X$ が推定した遅延の平均と標準偏差

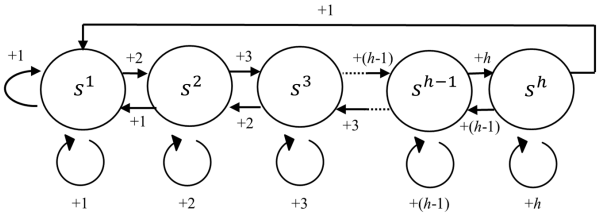

図 5.8: 崖っぷち問題: エージェントは $\{s^1, s^2, \ldots, s^h\}$ の中の各状態で「左」「停止」「右」から行動を選択し，状態 s^i にいるとき報酬 $+i$ を得る．

め難しい．

5.2.1　ノイズのない場合

ノイズがない場合は，エージェントは選んだ行動の通りに移動することができる．

事前の実験から各学習アルゴリズムのメタパラメータを次のように設定した．Sarsa(λ), dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の学習率を $\alpha = 0.3$，割引率を $\gamma = 0.2$，適格度トレースパラメータを $\lambda = 0.4$ とした．dSARSA(λ) は $\alpha = 0.03$ についても実験した．Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の行動方策をソフトマッ

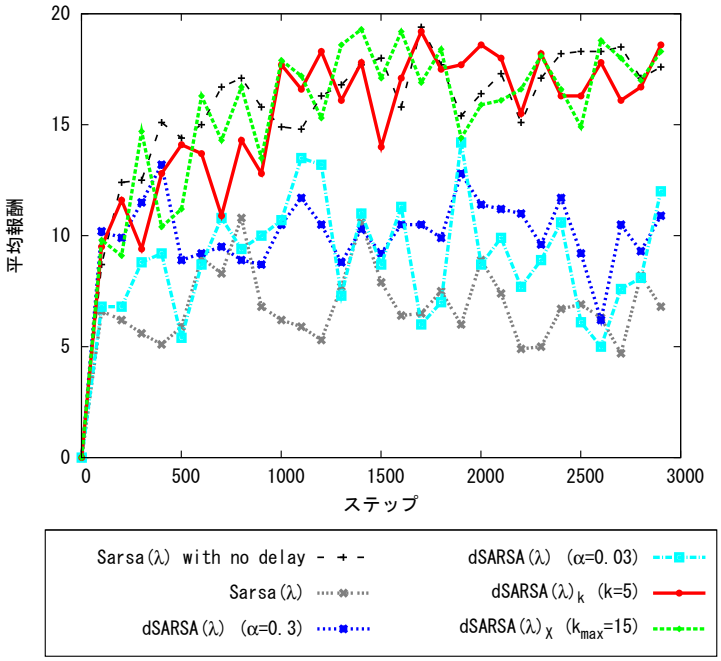

図 5.9: 崖っぷち問題を使った実験における，遅延がないときの Sarsa(λ) と，Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の比較

クス法とし，逆温度を $\beta = 0.2$ とした．真の遅延を $k^* = 5$ とし，dSARSA$(\lambda)_X$ については遅延の上限を $k_{\max} = 15$ とした．3,000 ステップを 10 回行い，各ステップで獲得する報酬の平均を求めた．

図 5.9，図 5.10 に実験結果を示した．比較として遅延がなかったときの Sarsa(λ) の平均報酬も示した．さらに，$k_{\max} = 5, 10, 15$ としたときの dSARSA$(\lambda)_X$ の推定した遅延の 10 回の平均と標準偏差を図 5.11 に示した．

実験結果は概ね “W-迷路” のときと同じで，真の遅延を与えられた dSARSA$(\lambda)_k$ は，他のアルゴリズムよりも高い平均報酬を獲得し，dSARSA$(\lambda)_X$ はこれと同等の性能を示した．

5.2.2 ノイズのある場合

ノイズがある場合は，エージェントは選んだ行動の通りに移動することができるとは限らない．エージェントは確率 0.9 で選んだ通りに移動できるが，それぞれ確率

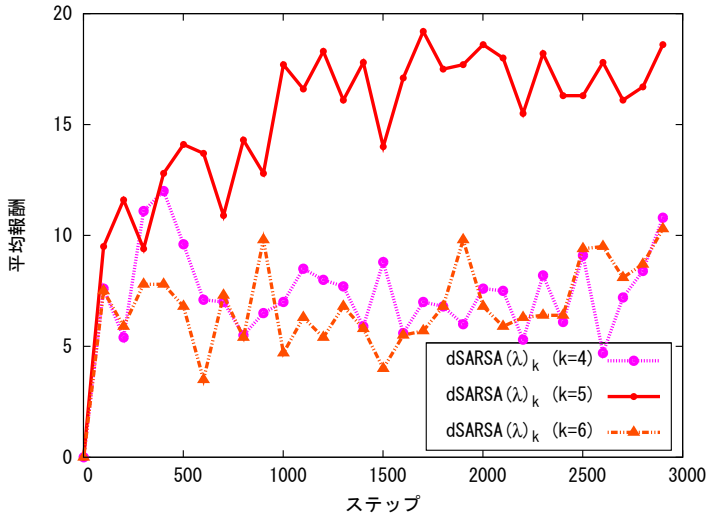

図 5.10: 崖っぷち問題を使った実験における，真の遅延 $k = 5$ と間違った遅延 $k = 4$, $k = 6$ をそれぞれ設定したときの dSARSA$(\lambda)_k$ の比較

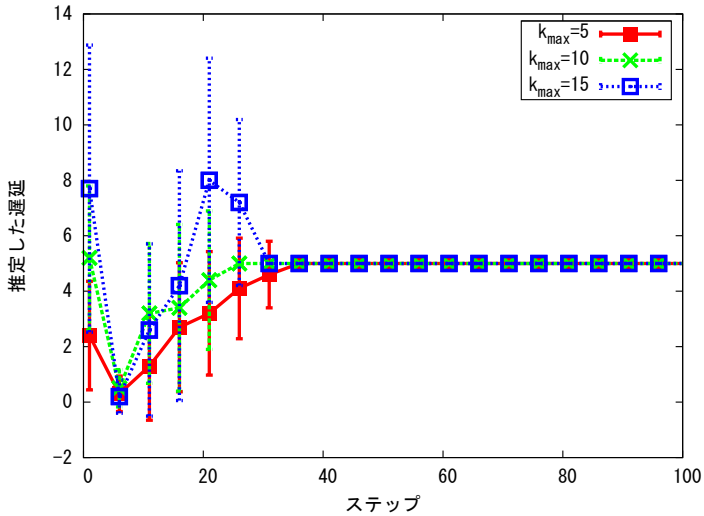

図 5.11: 崖っぷち問題を使った実験における，$k_{\max}$ を変化させたときの dSARSA$(\lambda)_X$ が推定した遅延の平均と標準偏差

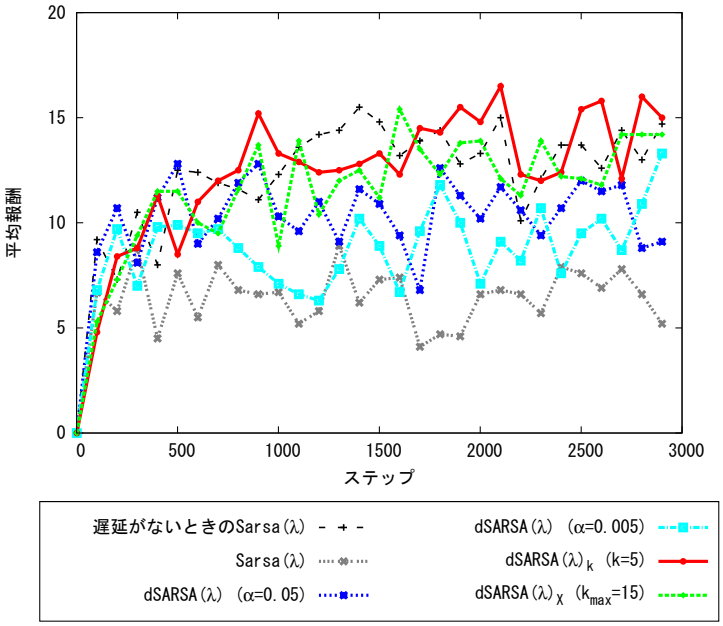

図 5.12: ノイズのある崖っぷち問題を使った実験における，遅延がないときの Sarsa(λ) と，Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の比較

0.05 で他の方向へ移動する．

事前の実験から各学習アルゴリズムのメタパラメータを次のように設定した．Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の学習率を $\alpha = 0.05$，割引率を $\gamma = 0.2$，適格度トレースパラメータを $\lambda = 0.4$ とした．dSARSA(λ) は $\alpha = 0.005$ についても実験した．Sarsa(λ)，dSARSA(λ)，dSARSA$(\lambda)_k$，dSARSA$(\lambda)_X$ の行動方策をソフトマックス法とし，逆温度を $\beta = 0.5$ とした．真の遅延を $k^* = 5$ とし，dSARSA$(\lambda)_X$ については遅延の上限を $k_{\max} = 15$ とした．3,000 ステップを 10 回行い，各ステップで獲得する報酬の平均を求めた．

図 5.12，図 5.13，図 5.14 に実験結果を示した．それぞれノイズがない場合の実験結果である，図 5.9，図 5.10，図 5.11 に対応している．

ノイズのために学習は難しくなっているが，それでも提案手法の結果は，遅延がないときの Sarsa(λ) の結果に近い．また，dSARSA$(\lambda)_X$ による遅延の推定も正確かつ高速であった．

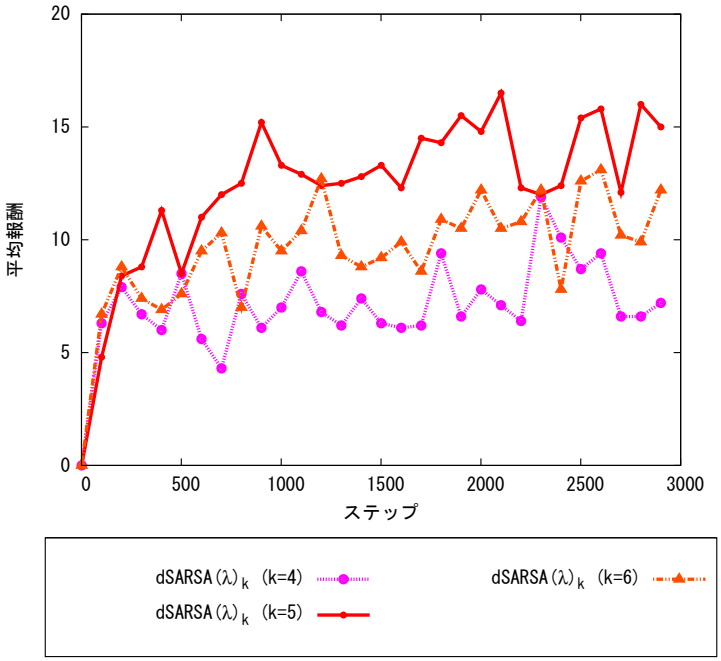

図 5.13: ノイズのある崖っぷち問題を使った実験における，真の遅延 $k = 5$ と間違った遅延 $k = 4$，$k = 6$ をそれぞれ設定したときの dSARSA$(\lambda)_k$ の比較

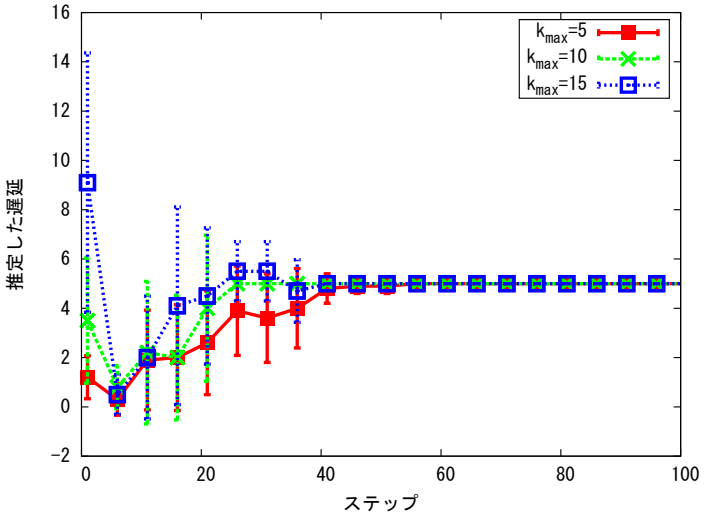

図 5.14: ノイズのある崖っぷち問題を使った実験における，$k_{\max}$ を変化させたときの dSARSA$(\lambda)_X$ が推定した遅延の平均と標準偏差

第6章

まとめと今後の課題

本稿では，遅延のあるマルコフ決定過程の下での強化学習を扱った．はじめに，定遅延マルコフ決定過程とこれに対する既存研究を紹介し，モデルフリー学習をもとにした dSARSA$(\lambda)_k$ を提案した．そして，計算量の大きいモデルベース学習の 1 つである MBS+R-max と比べて同程度の性能を示すことを実験的に確認した．次に，遅延の大きさが未知の場合について議論し，未知定遅延マルコフ決定過程を定義した．そして，未知定遅延マルコフ決定過程の理論的性質を示し，遅延を推定するアルゴリズム CPAS を提案した．さらに，遅延の推定と強化学習を同時に行う dSARSA$(\lambda)_X$ を提案した．dSARSA$(\lambda)_X$ は，真の遅延が未知であったときでも，真の遅延を与えたときの dSARSA$(\lambda)_k$ と実験的に同程度の性能を示すことを確認した．

今後の研究課題としては次が挙げられる．まず，理論的な視点からは，遅延を推定可能な未知定遅延マルコフ決定過程の必要十分条件やその条件を満たすならば常に遅延を推定できるアルゴリズムについての研究がある．さらに，本稿では時間と状態と行動が離散で状態と行動は有限，遅延は一定としたが，これらの条件を緩めた場合どのような問題が生じるのか，調べる必要がある．また，遅延を考慮することで実ロボットの学習の効率が実際に上がるのか検証したいと考えている．

参考文献

[1] P. Abbeel, A. Coates, M. Quigley, and A.Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*, Vol. 19, , 2007.

[2] R.I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, Vol. 3, pp. 213–231, 2003.

[3] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. Vol. 55, pp. 119–139, 1997.

[4] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, Vol. 58, No. 301, pp. 13–30, 1963.

[5] S.M. Kakade, et al. *On the sample complexity of reinforcement learning*. PhD thesis, University College London, 2003.

[6] K.V. Katsikopoulos and S.E. Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *Automatic Control, IEEE Transactions on*, Vol. 48, No. 4, pp. 568–574, 2003.

[7] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, Vol. 49, No. 2, pp. 209–232, 2002.

[8] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine learning*, pp. 1–33, 2011.

[9] L. Li. *A unifying framework for computational reinforcement learning theory*. PhD thesis, Rutgers, The State University of New Jersey, 2009.

[10] J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pp. 1–20, 2003.

[11] G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Cambridge University Department of Engineering, 1994.

[12] E. Schuitema, L. Busoniu, R. Babuška, and P. Jonker. Control delay in reinforcement learning for real-time dynamic systems: a memoryless approach. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3226–3231. IEEE, 2010.

[13] A.L. Strehl. *Probably approximately correct (PAC) exploration in Reinforcement Learning*. PhD thesis, Graduate School - New Brunswick Rutgers, The State University of New Jersey, 2007.

[14] A.L. Strehl, L. Li, E. Wiewiora, J. Langford, and M.L. Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888. ACM, 2006.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[16] R.S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, Vol. 216, 1990.

[17] I. Szita and C. Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the twenty-seventh international conference on machine learning*. Citeseer, 2010.

[18] T. Walsh, A. Nouri, L. Li, and M. Littman. Planning and learning in environments with delayed feedback. *Machine Learning: ECML 2007*, pp. 442–453, 2007.

[19] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

謝辞

学部，大学院の 3 年間にわたり，このような研究の場を与えて頂くとともに，本研究の直接的な指導教員として多くのご教示，ご鞭撻を賜りました東北大学 大学院情報科学研究科 篠原 歩 教授，ならびに成澤 和志 助教に厚く御礼申し上げます.

また，本論文審査の副審査員を務めて頂きました，東北大学 大学院情報科学研究科 田中 和之 教授，ならびに東北大学 電気通信研究所 石黒 章夫 教授には，御専門の立場から的確なご助言や貴重なご意見を賜りましたことを，心から御礼申し上げます.

そして，日頃の研究室での活動全般にわたりご支援頂いた東北大学 大学院情報科学研究科 篠原研究室の皆様に心から感謝申し上げます.

最後に，東北大学 大学院情報科学研究科 博士前期課程までという長い間，進学の機会を与えてくださり，最後まで信じて見守ってくださった家族，私生活においていつも温かく支えてくれた友人一同に，深く感謝いたします.